Status of the Claims

1. (currently amended) A method for executing a software application in a plurality of computers ~~computing nodes~~ having respective ~~node~~ hardware resources said hardware resources comprising a respective memory and a respective I/O device, wherein said ~~nodes~~ computers include a first computer ~~node~~ and a second computer ~~node~~ that intercommunicate over a network, ~~and~~ said ~~nodes~~ computers being operative to execute a virtual machine that runs under a guest operating system, comprising the steps of:

running at least a first virtual machine implementer and a second virtual machine implementer on said first computer ~~node~~ and said second computer ~~node~~, ~~respectively~~ using said respective memory; and

sharing said virtual machine between said first virtual machine implementer and said second virtual machine implementer using said respective I/O device in each of said first computer and said second computer to intercommunicate between said first computer and said second computer.

2. (currently amended) The method according to claim 1, further comprising the step of running said software application over said guest operating system, so that commands invoked by said software application are monitored or emulated by said first

virtual machine implementer and said second virtual machine implementer on said first computer ~~node~~ and said second computer ~~node~~, while said ~~node~~ hardware resources of said first computer ~~node~~ and said second computer ~~node~~ are shared by communication over said network.

3. (original) The method according to claim 1, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is a virtual machine monitor.

4. (original) The method according to claim 1, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is an emulator.

5. (currently amended) The method according to claim 1, wherein at least said first computer ~~node~~ comprises a first virtual node comprising a first physical CPU of said first computer ~~node~~ and a second virtual node comprising a second physical CPU of said first computer ~~node~~.

6. (original) The method according to claim 1, wherein said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first virtual machine and said second virtual machine have a plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on a first physical CPU and said second virtual machine implementer based on a second physical CPU, respectively.

7. (original) The method according to claim 6, and a first virtual node comprises said first physical CPU and said second physical CPU.

8. (original) The method according to claim 7, wherein said first virtual machine implementer virtualizes at least one of said virtual CPU's of said first virtual machine based on said first physical CPU and virtualizes at least one of said virtual CPU's in said second virtual machine based on said second physical CPU.

9. (currently amended) The method according to claim 1, further comprising the steps of:

providing a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer ~~node~~ and said second computer ~~node~~, respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer.

10. (currently amended) The method according to claim 9, further comprising the step of providing a virtual PCI controller for said management system to control a physical PCI controller in one of said ~~nodes~~ computers.

11. (currently amended) The method according to claim 9, further comprising the step of providing a virtual DMA controller

for said management system to control a physical DMA controller in one of said ~~nodes~~ computers.

12. (currently amended) The method according to claim 11, further comprising the steps of:

providing a virtual PCI controller to control a physical PCI controller in one of said ~~nodes~~ computers; and

during a bootup phase of operation scanning a device list with said virtual PCI controller to identify devices having on-board DMA controllers.

13. (currently amended) The method according to claim 1, further comprising the steps of:

with said first virtual machine implementer and said second virtual machine implementer maintaining mirrors of a portion of said respective memory that is used by said guest operating system in each of said ~~nodes~~ computers;

write-invalidating at least a portion of a page of said respective memory in one of said ~~nodes~~ computers; and

transferring a valid copy of said portion of said page to said one computer ~~node~~ from another of said ~~nodes~~ computers via said network.

14. (currently amended) A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions~~, when read by a computer,~~ cause a plurality of computers having respective hardware resources, said hardware resources comprising a

respective memory and a respective I/O device ~~the computer~~ to perform a method for executing a software application ~~in a plurality of computing nodes having node resources~~, wherein said ~~nodes~~ computers include a first computer ~~node~~ and a second computer ~~node~~ that intercommunicate over a network, and said ~~nodes~~ computers being operative to execute a virtual machine that runs under a guest operating system, comprising the steps of:

running at least a first virtual machine implementer and a second virtual machine implementer on said first computer ~~node~~ and said second computer ~~node~~, ~~respectively~~ using said respective memory; and

sharing said virtual machine between said first virtual machine implementer and said second virtual machine implementer using said respective I/O device in each of said first computer and said second computer to intercommunicate between said first computer and said second computer.

15. (original) The computer software product according to claim 14, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is a virtual machine monitor.

16. (original) The computer software product according to claim 14, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is an emulator.

17. (currently amended) The computer software product according to claim 14, wherein at least said first computer ~~node~~ comprises a first virtual node comprising a first physical CPU of said first computer ~~node~~ and a second virtual node comprising a second physical CPU of said first computer ~~node~~.

18. (original) The computer software product according to claim 17, wherein said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first virtual machine and said second virtual machine have a plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on said first physical CPU and said second virtual machine implementer based on said second physical CPU, respectively.

19. (original) The computer software product according to claim 18, wherein said plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on said first physical CPU and said second virtual machine implementer based on said second physical CPU, respectively.

20. (original) The computer software product according to claim 18, wherein said first virtual node comprises said first physical CPU and said second physical CPU.

21. (original) The computer software product according to claim 20, wherein said first virtual machine implementer virtualizes at least one of said virtual CPU's of said first

virtual machine based on said first physical CPU and virtualizes at least one of said virtual CPU's in said second virtual machine based on said second physical CPU.

22. (currently amended) The computer software product according to claim 14, wherein said computer is further instructed to perform the step of running said software application over said guest operating system, so that commands invoked by said software application are received by said first virtual machine implementer and said second virtual machine implementer on said first computer ~~node~~ and said second computer ~~node~~, while said ~~node~~ hardware resources of said first computer ~~node~~ and said second computer ~~node~~ are shared by communication over said network.

23. (currently amended) The computer software product according to claim 14, further comprising the steps of:

providing a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer ~~node~~ and said second computer ~~node~~, respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer and said second virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer and said second virtual machine implementer.

24. (currently amended) The computer software product according to claim 23, further comprising the step of providing a

virtual PCI controller for said management system to control a physical PCI controller in one of said computers ~~nodes~~.

25. (currently amended) The computer software product according to claim 23, wherein said computers ~~computer is~~ are further instructed to perform the step of providing a virtual DMA controller for said management system to control a physical DMA controller in one of said computers ~~nodes~~.

26. (currently amended) The computer software product according to claim 25, wherein said computers ~~computer is~~ are further instructed to perform the steps of:

providing a virtual PCI controller to control a physical PCI controller in one of said computers ~~nodes~~; and

during a bootup phase of operation scanning a device list with said virtual PCI controller to identify devices having on-board DMA controllers.

27. (currently amended) The computer software product according to claim 14, wherein said computers ~~computer is~~ are further instructed to perform the steps of:

with said first virtual machine implementer and said second virtual machine implementer maintaining mirrors of a portion of said respective memory that is used by said guest operating system in each of said computers ~~nodes~~;

write-invalidating at least a portion of a page of said respective memory in one of said computers ~~nodes~~; and

transferring a valid copy of said portion of said page to said one computer ~~node~~ from another of said computers ~~nodes~~ via said network.

28. (currently amended) A computer system for executing a software application, comprising:

a plurality of ~~computing nodes~~ computers having ~~node resources~~ respective hardware resources, said hardware resources comprising a respective memory and a respective I/O device, said ~~plurality of computing nodes~~ computers comprising at least a first computer ~~node~~ and a second computer ~~node~~;

a network connected to said first computer ~~node~~ and said second computer ~~node~~ providing intercommunication therebetween;

said first computer ~~node~~ and said second computer ~~node~~ being operative to execute a first virtual machine implementer and a second virtual machine implementer respectively using said respective memory, wherein a virtual machine is implemented concurrently by at least said first virtual machine implementer and said second virtual machine implementer; and

said computers ~~nodes~~ being operative to execute a guest operating system over said virtual machine, wherein said software application executes over said guest operating system, so that commands invoked by said software application are received by said first virtual machine implementer and said second virtual machine implementer on said first computer ~~node~~ and said second computer ~~node~~, while said ~~node~~ hardware resources of said first computer ~~node~~ and said second computer ~~node~~ are shared by communication over said network using said respective I/O device.

29. (original) The computer system according to claim 28, wherein said software application comprises a first software application and a second software application, said guest operating system comprises a first guest operating system and a second guest operating system, and said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first software application and said first guest operating system are associated with said first virtual machine, and said second software application and said second guest operating system are associated with said second virtual machine.

30. (currently amended) The computer system according to claim 29, wherein one of said computers ~~nodes~~ has a first physical CPU and a second physical CPU, and said first virtual machine implementer virtualizes a first virtual CPU in said first virtual machine based on said first physical CPU and virtualizes a second virtual CPU in said second virtual machine based on said second physical CPU.

31. (currently amended) The computer system according to claim 28, wherein at least said first computer ~~node~~ comprises a first virtual node and a second virtual node.

32. (currently amended) The computer system according to claim 31, wherein said first computer ~~node~~ comprises a first processor and a second processor, a first I/O device and a second I/O device, wherein said first I/O device is assigned to said

first processor, and said second I/O device is assigned to said second processor.

33. (currently amended) The computer system according to claim 28, further comprising a minimal operating system executing in each of said computers ~~nodes~~ to invoke said first virtual machine implementer and said second virtual machine implementer so that said first virtual machine implementer and said second virtual machine implementer control said computers ~~nodes~~.

34. (currently amended) The computer system according to claim 28, further comprising a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer ~~node~~ and said second computer ~~node~~, respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer and said second virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer and said second virtual machine implementer.

35. (currently amended) The computer system according to claim 34, further comprising a virtual PCI controller for said management system to control a physical PCI controller in one of said computers ~~nodes~~.

36. (currently amended) The computer system according to claim 34, further comprising a virtual DMA controller for said

management system to control a physical DMA controller in one of said computers ~~nodes~~.

37. (currently amended) The computer system according to claim 28, further comprising a memory management system ~~executing in at least one of said nodes~~ that maintains mirrors of a portion of said respective memory that is used by said guest operating system in each of said computers ~~nodes~~, wherein said memory management system write-invalidates at least a portion of a page of said respective memory in one of said computers ~~nodes~~; and transfers a valid copy of said portion of said page to said one computer ~~node~~ from another of said computers ~~nodes~~ via said network.

38. (new) The method according to claim 1, wherein said guest operating system consists of exactly one instance of a single guest operating system.

39. (new) The computer software product according to claim 14, wherein said guest operating system consists of exactly one instance of a single guest operating system.